ELSEVIER

Available online at www.sciencedirect.com

SCIENCE DIRECT®

Theoretical Computer Science 317 (2004) 1–11

Theoretical Computer Science

www.elsevier.com/locate/tcs

Preface

Three aspects of super-recursive algorithms and hypercomputation or finding black swans

Our engraved knowledge may bite into our thinking
certain errors that become well-nigh ineradicable.

Rogers MacVeagh and Thomas Costain "*Joshua*"

History of science and technology proves that the biggest advances come not from doing more and bigger and faster of what is already being done, but from new ideas, discoveries, and starting points. Hence this special issue concerns new ideas, discoveries, and metaphors in computer science. It is not about incremental improvements, but rather it presents the opportunities opened by these related notions: super-recursive algorithms and hypercomputation.

Together these new ideas, discoveries, constructions, and metaphors form a new computer science field, the *theory of super-recursive algorithms and hypercomputation*. It is a part of such established domain as the theory of algorithms, automata, and computation.

One main achievement of 20th century scientific thought is the theory of algorithms and computation. This theory studies abstract and real automata, computers and networks, computation and communication. In many ways, this theory is the central cornerstone for computer science. Many key accomplishments in the theory of algorithms and computation converge to the famous Church–Turing thesis, a statement determining the boundaries of algorithmic computations. The Church–Turing thesis has long been considered as the most fundamental law within computing. However, recent developments in the theory of algorithms allow overcoming limitations in the Church–Turing thesis. New mathematical models for algorithms and computation have appeared that extend prior theory in a manner similar to the way relativity theory and quantum mechanics went beyond Newtonian mechanics. These new models are more powerful than the classical recursive algorithm models, i.e., Turing machines, partial recursive functions, Lambda-calculus, and cellular automata.

Algorithms and automata that are more powerful than Turing machines are called super-recursive.

Computations that cannot be realized or simulated by Turing machines are called hypercomputations.

The new area of computer science called the theory of super-recursive algorithms and hypercomputation consists of several directions. The most important ones are

0304-3975/$ - see front matter
doi:10.1016/j.tcs.2003.12.001

listed here in chronological order: inductive computations and inference, computations and recursive functions with real numbers, interactive and concurrent computations, topological computations, infinite time computations, and neural networks with real number parameters. Each of these algorithmic and computational models supplies a new logic of computation. Together they extended the scope of computer science.

There were two conferences directed at problems of the theory of super-recursive algorithms and hypercomputation. The first, The *Hypercomputation Workshop* took place in London, May 24, 2000. It was a one-day workshop jointly supported by the Turing Project and the Philosophy Department of the University of Reading and organized by Jack Copeland and John Preston. The second, "*Beyond the Classical Boundaries of Computation*" was a two-day four-panel Session of the American Mathematical Society Meeting in San Francisco, May 2003.

Results in superrecursive algorithms and hyper-computation represent advances in the theoretical computer science domain. This special issue presents recent research in the area through thirteen papers that reflect different areas in the theory of super-recursive algorithms and hypercomputation. They are separated into four natural sections, going from more constructive to more abstract approaches:

I. Inductive and fuzzy algorithms and computation.
II. Infinite time and continuous models.
III. Logical and algebraic models.
IV. Methodological and philosophical aspects of super-recursive algorithms and hypercomputation.

Within the sections, the papers are ordered according to their submission date. Not all papers in this issue reflect opinions of the Editors of this issue on algorithms, computation, and their mathematical models. However, their publication supports scientific exchange, by exposing crucial problems, expressing different (sometimes contrasting) opinions, and suggesting non-trivial ideas for discussion. All papers have been duly refereed to comply with the high standards of the Journal "Theoretical Computer Science" and the best papers have been selected for this special issue.

In the paper of Peter Kugel "*Toward a Theory of Machine Intelligence*", super-recursive algorithms and hypercomputation are studied in the context of problems of natural and artificial intelligence. Different algorithmic models of intelligence based on recursive and inductive computational modes are considered and compared with respect to their cognitive abilities. Currently, the theory of super-recursive algorithms and hypercomputation uses two terms, limiting and inductive computation, to indicate a stabilizing computational process. The term *limiting computation* first appeared in the pioneering paper of Gold [10]. However, this term gives a wrong impression. It implicitly implies that such processes demand infinite time and number of computational steps to obtain a result. Contrary to this, any computation of this type gives its result after some finite time interval and demands a finite number of simple steps [7]. At the same time, the computational process in question works like induction by accumulating intermediary results to produce the final result at some step. Thus, the term *inductive computation* is more accurate.

In 1950, Turing wrote that computers could be programmed to behave intelligently, but only if we enable them to do more than compute. In this paper, Kugel uses a mathematical model of intelligence to clarify Turing's suggestion. Obtained results bring him to the conclusion that it is probably better to try to provide that "more" by developing new software (that will allow computers to run super-recursive algorithms) than by developing new hardware (that will allow us to turn computers into hypercomputers).

However, Kugel does not stop obtaining interesting results for inductive computation. He formulates several prospective directions for the development of artificial intelligence and its application to computer science and software technology.

In his paper "*Characterizing the Super-Turing Computing Power and Efficiency of Classical Fuzzy Turing Machines*," Jiri Wiedermann introduces and studies accepting fuzzy Turing machines with the same structure as conventional Turing machines, but in which each step of computation is performed with some truth degree. This concept formalizes the idea of Zadeh of a fuzzy algorithm. If we take all computations of fuzzy Turing machines, we obtain a new class of non-deterministic Turing machines. It is known that these machines have the same computing and accepting power as deterministic Turing machine. However, a distinction between different levels of the acceptation truth degree, we essentially extend the power of these machines. They become models of super-recursive algorithms, hence are capable of performing hypercomputation. The author proves that fuzzy languages accepted by these machines with a computable t-norm correspond exactly to the union $\Sigma_1^0 \cup \Pi_1^0$ of recursively enumerable languages and their complements. The second main result of the paper states that the complexity class of polynomially time-bounded computations of such machines coincides with the union NP $\cup$ co-NP from the first level of the polynomial hierarchy.

In his paper "*Algorithmic complexity of recursive and inductive algorithms*" Mark Burgin continues his research of inductive computations. There are three main types of inductive processes:

Cognitive induction when the problem is either to find some properties of a given object/system or to find an object/system that has some given properties. Conventional methods of inductive inference give examples of cognitive induction.

Constructive induction [13] consists of a search when the process of finding an object or system with given properties is accomplished by a search domain transformation. Inductive inference with optimization of the search domain gives examples of cognitive induction.

Inductive construction involves a problem of building an object or system that satisfies some given properties. Computations by inductive Turing machines are a kind of inductive construction.

The paper "*Algorithmic complexity of recursive and inductive algorithms*" compares recursive algorithms such as Turing machines with such super-recursive algorithms as inductive Turing machines (ways to realize inductive computations). This comparison is made in a general setting of dual complexity measures such as Kolmogorov or algorithmic complexity. To make adequate comparison, it becomes necessary to reconsider the standard axiomatic approach to complexity of algorithms. The new approach allows the author to achieve a more adequate representation of static system complexity in the axiomatic context. This result demonstrates that for solving many problems

inductive Turing machines have much lower complexity than Turing machines and other recursive algorithms. This affirms that inductive Turing machines are not only more powerful, but also more efficient than Turing machines.

Mark Burgin and Allen Klinger, in their paper " Experience, Generations, and Limits in Machine Learning," study problems of machine learning in a setting of super-recursive algorithms. They extend traditional models of machine learning beyond their one-level structure by introducing previously obtained problem knowledge into the algorithm or automaton involved. Some authors studied models that utilize some kind of predetermined knowledge, having a two-level structure. However, even in this case, the model has not reflected the source and inherited properties of predetermined knowledge. In society, knowledge is often transmitted from previous generations. The aim of this paper is to construct and study algorithmic models of learning processes that utilize predetermined or prior knowledge. The models use recursive, subrecursive, and super-recursive algorithms. Predetermined knowledge includes: a text description, activity rules (e.g., for cognition), and specific structured individual or social memory. Algorithmic models represent these three forms as separate structured processing systems: automata with (1) advice; (2) structured program; and (3) structured memory. That yields three basic models for learning systems: polynomially bounded Turing machines, Turing machines, and inductive Turing machines of the first order. It is demonstrated that only inductive Turing machines allow a learner to increase cognitive power through transmitted knowledge utilization.

In his paper "*Hypercomputation with Quantum Adiabatic Processes*," Tien D Kieu discusses an alternative approach to quantum computation. The standard approach is based on linear superposition and entanglement that give quantum computation its power and efficiency. However, computational power of abstract quantum computers (real quantum computers still do not exist) does not surpass the power of ordinary Turing machines [8]. Tien D Kieu suggests a quantum algorithm based on the Quantum Adiabatic Theorem. He shows how this algorithm solves Hilbert's 10th problem (although it is proved unsolvable by standard mathematical methods). Since standard mathematical methods can achieve no more than an ordinary Turing machine, this shows that adiabatic quantum computations are theoretically more powerful than standard quantum computations.

It is interesting that Kieu suggests that the halting of a quantum universal Turing machine is highly problematic. This brings us to necessity of modeling quantum computers with super-recursive algorithms, such as inductive and limit Turing machines.

In his paper "*Super-Tasks, Accelerating Turing Machines and Uncomputability*," Oron Shagrir discusses problems of accelerating Turing machines, abstract devices that have the same computational structure as Turing machines, but can perform super-tasks. The author argues that performing super-tasks makes accelerating Turing machines essentially different in comparison with ordinary Turing machines.

The reason is that to have a full description of an algorithm or abstract automaton/machine A, we must:

1. Describe the structure of A (*the static structure*).
2. Describe how each step of computation of A is performed (*the dynamic structure*).
3. Describe how a result is obtained (*the end structure*).

This implies the following consequences for accelerating machines:

1. Accelerating machines may have the static structure of an ordinary Turing machine. In this case, it is possible to call them accelerating Turing machines.
2. Accelerating Turing machines can be different from ordinary Turing machines because they have different end structure. So, they are not ordinary Turing machines.
3. However, to be precise, it is necessary to describe explicitly the end structure of accelerating Turing machines. There are different ways to define the end structures. One possibility is to use the end structure of an inductive Turing machine or limiting partial recursive function (e.g., [7]). Another possibility is to use the end structure of an infinite time Turing machine (e.g., [11]). One more possibility is to use the end structure of a limit Turing machine [4]. Moreover, even for an accelerating Turing machine, it is possible to define such end structure that will make it equivalent to an ordinary Turing machine.

Bruce MacLennan in his paper "*Natural computation and non-Turing models of computation*," gives a critique of ordinary Turing machines and suggests a new approach called natural computation. He defines *natural computation* as computation occurring in nature or inspired by that in nature. This definition makes no commitment as to whether discrete or continuous models are preferable. This is an empirical issue, and no doubt different instances of natural computation will require different sorts of models. Since the theory of discrete computation is more developed, MacLennan focuses on continuous models, and in particular on *field computation*.

One of the shortcomings of the traditional model of computation is that it is oriented toward potential unbounded complexity, that is, how utilization of some resource (typically time or space) grows with the size of the input. Such analysis is less relevant in the context of natural computation, since the size of the input is generally fixed (e.g., by the structure or anatomy of a sensory system). Natural computational models allow one to use other criteria for comparison of algorithms: *speed of response*, *generality of response*, *flexibility in response to novelty*, *adaptability*, *tolerance to noise, error, faults, and damage*.

While other approaches to continuous and topological computations (Shannon, Pour-El, Rubel, Moore, Blum et al., Burgin) give different models of continuous and topological computations, MacLennan develops foundations for the theory of continuous and topological computations and algorithms, utilizing an axiomatic approach. As a result, he achieves higher level of theory than other authors who worked in this area. For a general theory of computations and algorithms, axiomatic systems are developed in [3].

One of the main axioms (Postulate 4) suggested by MacLennan for continuous and topological computations states that mappings between image spaces, that is, elementary computational operations, are continuous. It is possible to extend this axiom assuming that such operations are only fuzzy continuous [5]. This better reflects properties of natural computation. For instance, the ability to adapt gradually to novelty implies that physical representations of natural computational processes are at least partially continuous. In addition, fuzzy continuity of computational operations allows one to include continuous and discrete computations in the same axiomatic setting.

In his paper "*Continuous time computation with restricted integration capabilities*," Manuel Lameiras Campagnolo develops further the recursion theory on real numbers, the analog counterpart of recursive function theory. It is a model of continuous-time computation inspired by the models of classical physics. In this theory, the discrete operations of standard recursion theory are replaced by operations on continuous functions such as composition and various forms of differential equations like indefinite integrals, linear differential equations and more general Cauchy problems. Classes of real recursive functions are defined in a manner similar to the standard recursion theory with the aim to study their complexity. Several classes of real recursive functions are characterized in terms of space complexity. In particular, it is demonstrated that hierarchies of real recursive classes closed under restricted integration operations are related to the exponential space hierarchy.

In his paper "*The Modal Argument for Hypercomputing Minds*," Selmer Bringsjord discusses the problem whether the human mind hypercomputes, or merely computes. There are many informal arguments from Gödel, Lucas, Penrose and others for the view that, in light of incompleteness theorems, the human mind has powers exceeding those of Turing machines and their equivalents. All these arguments fail. Their flaws have been repeatedly exposed in the literature. Bringsjord gives herein a formal *modal* argument showing that the approach that equates mind with a computer is false. This allows him to state that minds *are* in fact hypercomputers. After this, the author refutes several objections to this statement and considers different models of super-recursive algorithms and hypercomputation.

The paper "*Hypercomputation by Definition*" by Benjamin Wells discusses algorithmic problems of such algebraic structures as pseudorecursive varieties of semigroups and such logical structures as equational theories of semigroups. Following Alfred Tarski, the author considers as decidable non-recursive equational theories in which any subtheory generated by formulas with a finite number of variables is recursive. Then a process that makes such theories and corresponding varieties decidable is called hypercomputation by definition. Wells discusses several approaches and mathematical models that make such theories decidable, suggesting directions for future research.

In her paper "*The Concept of Computability*," Carol Cleland attacks Turing machines from two sides. On one hand, she demonstrates that Turing machine is a far-reaching idealization of physical systems, and it surpasses such systems in different aspects. On the other hand, Cleland suggests a possibility of physical computing devices that can do more than Turing machines, thus realizing hypercomputation. Refining the concept of computation is the aim and base of Cleland's analysis. In developing this analysis Cleland debates and reconsiders the main principles of the contemporary theory of algorithms and computation. She brings us to one main conclusion: it is necessary to continue logical and methodological analysis of computer science foundations to stimulate its growth, and choose the right direction for its development.

The central question behind the reasoning in Cleland's paper involves how it is possible that mathematics, in her case, it is the mathematical structure that is called specifically a Turing machine, being so exact and perfect, can be so successfully applied to the imprecise and imperfect physical world—specifically how can an ideal and formalized theory influence real computers and computation. This question excited many

outstanding thinkers, including Albert Einstein and Eugene Wigner. One answer is that mathematics takes its objects from reality, developing images of real things by means of generalization, idealization, and abstraction (cf., for example, [1] or [12]). However, the bulk of mathematical knowledge has been created inside mathematics without any reference to reality. Only later, many of the mathematical concepts and methods have found applications in the real world. This brings us to another answer to the crucial question about the miraculous efficiency of mathematics. This approach synthesizes the teaching of Plato on ideas with that of Aristotle on forms. It results in a new theory of structures called structurology [6]. From the perspective of structurology, structures of things and processes determine the essence of these things and processes. Mathematics is a formalized study of various structures, which includes many structures from the real world [2]. This is the source of mathematics power.

This is the reason why, in spite of all shortcomings and critique, ordinary Turing machines will continue to be useful in computer science like Newton's laws in physics. The remaining necessity is to determine clear boundaries for applying of such machines to be models of real systems and processes. The methodological analysis conducted in papers of Cleland and MacLennan helps to achieve this goal.

In the paper "*The Problem of Induction and the Problem of Computation*," Kevin T. Kelly compares processes of computation in computer science, formal reasoning in mathematics, and induction in empirical sciences. In formal reasoning and computability, finding the right answer is usually understood to imply *halting* in a finite time with a correct result. Empirical science, on the other hand, is notoriously fallible, because there is no time by which science can infallibly announce that the law is true. At best, the truth is arrived at, eventually, with no bell announcing when it has succeeded. Arguments are given that uncomputable formal problems are intuitively, mathematically, and methodologically analogous to general empirical questions, warranting a similar, fallible attitude in the formal domain. Kelly argues that a version of Ockham's razor (a preference for the simplest answer compatible with experience) is advantageous in both domains. The analogies imply that when halting with a correct answer is algorithmically infeasible, we may drop the halting requirement in formal reasoning. That is what is done in empirical reasoning. This idea results in a notion of "hypercomputation" based entirely on classical computational models and on methods parallel to attitudes long familiar in the empirical domain.

Jack Copeland, in his paper "*Hypercomputation: Philosophical Issues*," discusses philosophical issues of hypercomputation. At first, he describes some elementary models of hypercomputation. These abstract machines serve to make the point that computability is a relative notion, not an absolute one. All computation takes place relative to some set or other of capacities, richer or poorer. The capacities specified by Turing in 1936 occupy no privileged position. Then Copeland analyzes some objections to the possibility of hypercomputation and considers in more detail Turing's oracle machines or *o*-machines. The paper concludes with a discussion on some exegetical issues concerning the writing of Turing and Church.

Emergence of a diversity of computational and hypercomputational models brings us to an important question. It is whether we can do with computers what it is possible to do with models. There is no simple answer to this question. The reality here is much

more complex. Now there are several approaches in this direction. However, without a sound mathematical base people, even good experts in computer science, cannot make distinctions nor choose directions better suited for research and application. Hence, it is important not only to go beyond the Church–Turing thesis, but to do it realistically and in a grounded way. For example, Turing machines with oracles took computations far beyond the Church–Turing thesis. But without specific restriction on oracles, that approach also was beyond reason.

The inability to make distinctions fosters misunderstanding and misconceptions: everything seems the same, although some models are unreasonable and in many cases, not realizable, while it is possible to embody other models in technological devices. At the same time, the third class of models may be a useful tool for investigation of natural system, such as the brain. We can compare the situation with existing models of super-recursive algorithms and hypercomputation to one when people do not and cannot distinguish between works of Ziolkowski that gave a theory for space flights and novels of Jules Verne that described such flights in the form of science fiction.

Mathematics can help us to gain understanding of the real situation. In many areas of mathematics, including theory of algorithms and computation, it models reality. The mathematical approach can likewise supply means to evaluate correctly super-recursive algorithms and models of hypercomputation.

First, super-recursive algorithms are mathematical models of computations and computers, although they give only approximate ideal representation of their object domain. Einstein stated that this in a paradoxical extreme form: as far as the laws of mathematics refer to reality, they are not certain; and as far as they are certain, they do not refer to reality. For instance, if we take Turing machine, itself the most popular model for computation, we see that no real device can have an infinite memory, even potentially. Thus, Turing machine is an idealization of real computers. However, this abstract idealization allows us to find many important properties of real computers

This is similar to the situation in science where mathematics often leads to discovery of new fundamental laws of nature, via approximate images of reality.

Here is an example from physics: The first Kepler's law states:

The planets move round the Sun in ellipses.

When people say or write that the real orbits of planets are ellipses, this is only approximation to real movement, which is much more complex as planets are influenced not only by the Sun, but also by other planets and, to a less extent, by other cosmic bodies.

Second important issue of the theory of super-recursive algorithms and hypercomputation is that computations, computers, and networks can be considered to be realizations of super-recursive algorithms. Hence, it is possible to ask to what extent it is possible to realize features of a given class or model of super-recursive algorithms.

Third, super-recursive algorithms provide the core for mathematical models of utilization of computations, computers, and networks. These questions are only at the beginning of their exploration, both in practice and in theory.

Nevertheless, the situation with super-recursive algorithms is not so simple. James Gleick in his book "Chaos" [9] cites a physicist at the Georgia Institute of Technology,

Joseph Ford, who quoted Tolstoy:

"I know that *most men*, including those at ease with problems of greatest complexity, *can seldom accept even the simplest and most obvious truth* if it be such as would oblige them to admit the falsity of conclusions which they have delighted in explaining to colleagues, which they have proudly taught to others, and which they have woven, thread by thread into the fabric of their life."

So, it is not a surprise that some people object to super-recursive algorithms and hyper-computation (e.g., [14]). Their main argument is that many have attempted but nobody accomplished such models that go beyond the Church–Turing thesis. We have many examples refuting this argument.

One of the most renowned is related to the famous model of a true empirical proposition that is attributed to Aristotle and is obtained by observation and induction:

All swans are white.

Europeans had believed in this until they came to Australia where they found black swans and understood that the statement of Aristotle is false.

To consider these issues, we have to make a distinction between different types and kinds of super-recursive algorithms. Analysis of the current situation in the theory of super-recursive algorithms and hypercomputation shows different aspects of this theory.

First, the variety of models of super-Turing computations contains models that are realistic from the point of view of their computer implementation, but they are not analogues computations or "black box" models. To find what is more realistic, what is less realistic, and what is only a useful abstraction, we need a more detailed, thorough and extended analysis of all models of super-Turing computations.

Second, there are many models of hypercomputation that are not algorithms, but only algorithmic or computational schemes. Examples are models of infinite time and analogue computations, as well as Turing machines with unrestricted oracles.

Third, it is incorrect to suggest that these schemes are useless. Even ignoring their utility for the theory of algorithms and computation, we can state that all models of hypercomputations or super-Turing computations might be useful for information technology if they are *correctly* applied to problems of technology.

Now the majority of mathematicians, computer scientists, and philosophers who research problems of computation and algorithms believe that the Church–Turing thesis is true and Turing machines restricts all possible kinds of computability. However, we have many examples from the history of mathematics, science, and even logic that many beliefs that were promoted to the range of empirical laws were later disproved.

However, not only general statements, but also authoritative theories have been changed after new discoveries. Newton's mechanics gives us one of the most impressive examples of such situations in science. For several centuries scientists believed in universality and omnipotence of Newton's mechanics. However in the 20th century, relativity theory and quantum mechanics demonstrated that Newton's laws, although being very important, are restricted only to macroscopic phenomena. In the microworld and in the world of high velocities other laws are valid.

In a similar way, it is possible to compare some super-recursive algorithms to relativity theory, which gives better models for high velocities, while other super-recursive algorithms resemble quantum mechanics, which gives better models for microscopic processes.

The main problem of the theory of super-recursive algorithms and hypercomputation is that now there are several approaches in this direction, but without sound mathematical base people, even good experts in computer science, cannot make distinctions for these directions. However, it is important not only to go beyond the Church–Turing thesis, but also to do this realistically and in a grounded manner. For example, Turing machines with arbitrary oracles take computations far beyond the Church–Turing thesis. However, only adequate restrictions on the oracle make such models useful and/or realistic.

Disability to make distinctions implies misunderstanding and misconceptions: everything seems the same, although for some it is all unreasonable and non-realizable, while for others it is all important, while trivial. The situation is similar to one when people do not and cannot make distinctions between mathematical and logical works of Charles Lutwidge Dodgson (Lewis Carrol), his famous books about Alice, and his poetry.

Making necessary distinctions in the variety of super-recursive algorithms and models of hypercomputation, we come to the following conclusion. Some of these models and algorithms reflect essential peculiarities of modern computers, embedded devices, and networks, providing relevant models for such systems. Some of these models and algorithms give ideas and methods for weighty advancement of computer technology. At the same time, others are only theoretical constructions that are aimed at the development of strictly theoretical areas of computer science and mathematics.

Mark Burgin, Allen Klinger
Department of Computer Science,
University of California,
Los Angeles, *405 Hilgard Ave.*,
Los Angeles, *CA 90095*, *USA*
E-mail address: mburgin@math.ucla.edu

References

[1] I. Asimov, The Words of Science and the History behind Them, Houghton Mifflin, Boston, MA, 1959.
[2] N. Bourbaki, The architecture of mathematics, Amer. Math. Monthly 57 (1950) 221–232.
[3] M. Burgin, Algorithms and algorithmic problems, Programming 4 (1985) 3–14 (Programming and Comput. Software 11(4)) (translated from Russian).
[4] M. Burgin, Universal limit Turing machines, Notices Russian Acad. Sci. 325 (1992) 654–658 (translated from Russian).
[5] M. Burgin, Neoclassical analysis: fuzzy continuity and convergence, Fuzzy Sets and Systems 75 (1995) 291–299.
[6] M. Burgin, On the Nature and Essence of Mathematics, Ukrainian Academy of Information Sciences, Kiev, 1998 (in Russian).
[7] M. Burgin, How we know what technology can do, Comm. ACM 44 (11) (2001) 82–88.

[8] D. Deutsch, A. Ekert, R. Lupacchini, Machines, logic and quantum physics, Bull. Symbolic Logic 6 (3) (2000) 265–283.
[9] J. Gleick, Chaos: Making a New Science, Cardinal, London, 1989.
[10] E.M. Gold, Limiting recursion, J. Symbolic Logic 30 (1) (1965) 28–46.
[11] J.D. Hamkins, A. Lewis, Infinite time Turing machines, J. Symbolic Logic 65 (2000) 567–604.
[12] A.N. Kolmogorov, Mathematics, in: Y.V. Prohorov (Ed.), Mathematical Encyclopedic Dictionary, Soviet Encyclopedia, Moscow, 1988, pp. 7–38.
[13] R.S. Michalski, J. Wnek, Constructive induction an automated design of knowledge representation spaces for machine learning, Reports of the Machine Learning and Inference Laboratory, MLI 93-11, School of Information Technology and Engineering, George Mason University, Fairfax, VA, November 1993.
[14] C. Teuscher, M. Sipper, Hypercomputation: hype or computation?, Comm. ACM 45 (8) (2002) 23–24.
[15] A.M. Turing, Computing machinery and intelligence, Mind 59 (N.S. 236) (1950) 433–460.